|  |  |
| --- | --- |
| **BUBBLE SORT**  **#include <stdio.h>**  **#include <stdlib.h>**  **void main()**  **{**  **int a[100],n, element;**  **int i, j, flag = 1;**  **printf("ENTER THE NO OF ELEMENTS \n");**  **scanf("%d",&n);**  **printf("ENTER ELEMENTS TO SORT\n");**  **for(i = 0; i < n ; i++)**  **{**  **scanf("%d", &a[i]);**  **}**    **printf("BEFORE SORTING\n");**  **for(i = 0; i < n; i++)**  **{**  **printf("%d ", a[i]);**  **}**  **printf("\n");**    **for(i = 0; i<5; i++)**  **{**  **flag = 0;**  **for(j = 0; j < 5-i; j++)**  **{**  **if(a[j] > a[j+1])**  **{**  **element = a[j];**  **a[j] = a[j+1];**  **a[j+1] = element;**  **flag = 1;**  **}**  **}**  **}**    **printf("AFTER SORTING\n");**  **for(i = 0; i < n; i++)**  **{**  **printf("%d ", a[i]);**  **}**  **}** | **SELECTION SORT**  **#include <stdio.h>**  **#include <stdlib.h>**  **void main()**  **{**  **int a[100],n, element;**  **int i, j, flag = 1;**  **printf("ENTER NO OF ELEMENTS\n");**  **scanf("%d",&n);**  **printf("ENTER ELEMENTS TO SORT\n");**  **for(i = 0; i < n ; i++)**  **{**  **scanf("%d", &a[i]);**  **}**    **printf("BEFORE SORTING\n");**  **for(i = 0; i < n; i++)**  **{**  **printf("%d ", a[i]);**  **}**  **printf("\n");**    **for(i = 0; i<5; i++)**  **{**  **flag = i;**  **for(j = 0; j < 5-i; j++)**  **{**  **if(a[j] > a[j+1])**  **{**  **flag=j;**  **element = a[j];**  **a[j] = a[j+1];**  **a[j+1] = element;**  **}**  **}**  **}**    **printf("AFTER SORTING\n");**  **for(i = 0; i < n; i++)**  **{**  **printf("%d ", a[i]);**  **}**  **}** |

**EXPERIMENT 1 + 2 + 3 : SORTING TECHNIQUES**

**INSERTION SORT**

**#include<stdio.h>**

**void main()**

**{**

**int a[100],n,i;**

**printf("ENTER NO OF ELEMENT \n");**

**scanf("%d",&n);**

**printf("ENTER ELEMENTS TO SORT\n");**

**for(i = 0; i < n ; i++)**

**{**

**scanf("%d", &a[i]);**

**}**

**printf("BEFORE SORTING\n");**

**for(i = 0; i < n; i++)**

**{**

**printf("%d ", a[i]);**

**}**

**printf("\n");**

**int i,j,temp;**

**for(i=1;i<n;i++)**

**{**

**temp=a[i];**

**j=i-1;**

**while(temp<a[j] && j>=0)**

**{**

**a[j+1]=a[j];**

**j--;**

**}**

**a[j+1]=temp;**

**}**

**}**

**printf("AFTER SORTING\n");**

**for(i = 0; i < n ; i++)**

**{**

**printf("%d ", a[i]);**

**}**

**}**

**MERGE SORT**

|  |  |
| --- | --- |
| **#include<stdio.h>**  **#include<stdlib.h>**  **#define SIZE 100**  **void merge(int a[],int low, int mid,int high)**  **{**  **int i,j,k=0;**  **i=low;**  **j=mid+1;**  **int c[high];**  **while(i<=mid && j<=high)**  **{**  **if(a[i]<a[j])**  **{**  **c[k]=a[i];**  **i++;**  **}**  **else**  **{**  **c[k]=a[j];**  **j++;**  **}**  **k++;**  **}**  **while(i<=mid)**  **{**  **c[k]=a[i];**  **i++;**  **k++;**  **}**  **while(j<=high)**  **{**  **c[k]=a[j];**  **j++;**  **k++;**  **}**  **for(i=low,k=0;i<=high;i++,k++)**  **{**  **a[i]=c[k];**  **}**  **}**  **void mergesort(int a[],int low,int high)**  **{**  **int mid;**  **if(low<high)**  **{**  **mid=(low+high)/2;**  **mergesort(a,low,mid);**  **mergesort(a,mid+1,high);**  **merge(a,low,mid,high);**  **}**  **}** | **void main()**  **{**  **int a[SIZE],i,j,n;**  **printf("ENTER NO OF ELEMENT \n");**  **scanf("%d",&n);**  **printf("ENTER ELEMENTS TO SORT\n");**  **for(i = 0; i < n ; i++)**  **{**  **scanf("%d", &a[i]);**  **}**  **printf("BEFORE SORTING\n");**  **for(i = 0; i < n; i++)**  **{**  **printf("%d ", a[i]);**  **}**  **printf("\n");**  **mergesort(a,0,n-1);**  **printf("AFTER SORTING\n");**  **for(i = 0; i < n ; i++)**  **{**  **printf("%d ", a[i]);**  **}**  **}** |

**QUICK SORT**

|  |  |
| --- | --- |
| **#include<stdio.h>**  **#include<stdlib.h>**  **#define SIZE 100**  **void swap(int \*a,int \*b)**  **{**  **int temp;**  **temp=\*a;**  **\*a=\*b;**  **\*b=temp;**  **}**  **int partition(int a[],int low,int high)**  **{**  **int pivot,i,j;**  **pivot=a[low];**  **i=low;**  **j=high;**  **while(pivot >= a[i])**  **{**  **i++;**  **}**  **while(pivot<a[j])**  **{**  **j--;**  **}**  **if(i<j)**  **{**  **swap(&a[i],&a[j]);**  **}**  **swap(&a[low],&a[j]);**  **return j;**  **}**  **void quicksort(int a[],int low,int high)**  **{**  **int p;**  **if(low<high)**  **{**  **p=partition(a,low,high);**  **quicksort(a,low,p-1);**  **quicksort(a,p+1,high);**  **}**  **}** | **void main()**  **{**  **int a[SIZE],i,j,n;**  **printf("ENTER NO OF ELEMENT \n");**  **scanf("%d",&n);**  **printf("ENTER ELEMENTS TO SORT\n");**  **for(i = 0; i < n ; i++)**  **{**  **scanf("%d", &a[i]);**  **}**  **printf("BEFORE SORTING\n");**  **for(i = 0; i < n; i++)**  **{**  **printf("%d ", a[i]);**  **}**  **printf("\n");**  **quicksort(a,0,n-1);**  **printf("AFTER SORTING\n");**  **for(i = 0; i < n ; i++)**  **{**  **printf("%d ", a[i]);**  **}**  **}** |

**EXPERIMENT 4 : MINMAX DIVIDE AND CONQUER**

**#include<stdio.h>**

**#include<stdlib.h>**

**typedef struct mm**

**{**

**int min;**

**int max;**

**}mm;**

**mm minmax(int a[],int low,int high)**

**{**

**mm p,q,res;**

**int mid;**

**if(low+high>=0)**

**{**

**//ONLY ONE INPUT**

**if(low==high)**

**{**

**res.min=a[low];**

**res.max=a[low];**

**}**

**//ONLY TWO INPUTS**

**else if (low+1==high)**

**{**

**if(a[low]>=a[high])**

**{**

**res.min=a[high];**

**res.max=a[low];**

**}**

**else**

**{**

**res.min=a[low];**

**res.max=a[high];**

**}**

**}**

**//MULTIPLE INPUTS**

**else**

**{**

**mid=(low+high)/2;**

**p=minmax(a,low,mid);**

**q=minmax(a,mid+1,high);**

**if(p.min<q.min)**

**res.min=p.min;**

**else**

**res.min=q.min;**

**if(p.max>q.max)**

**res.max=p.max;**

**else**

**res.max=q.max;**

**}**

**}**

**return res;**

**}**

**void main()**

**{**

**mm res;**

**int n,i,a[10];**

**printf("ENTER THE SIZE OF THE ARRAY ");**

**scanf("%d",&n);**

**for(i=0;i<n;i++)**

**{**

**printf("Enter the %d element of the array : ",i+1);**

**scanf("%d",&a[i]);**

**}**

**res=minmax(a,0,n-1);**

**printf("The MINIMUM ELEMENT OF THE ARRAY IS %d\n",res.min);**

**printf("THE MAXIMUM ELEMENT OF THE ARRAY IS %d\n",res.max);**

**}**

**EXPERIMENT 5 – FRACTIONAL KNAPSACK**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct bag**

**{**

**float weight,ratio,profit,frac;**

**int no;**

**};**

**void main()**

**{**

**struct bag s[10],temp;**

**int i,j,n;**

**float limit,fractionalprofit,totalprofit;**

**//TAKE IN INFORMATION**

**printf("ENTER THE NUMBER OF ITEMS: ");**

**scanf("%d",&n);**

**printf("Enter the limit : ");**

**scanf("%f",&limit);**

**for(i=0;i<n;i++)**

**{**

**s[i].no=i;**

**printf("WEIGHT OF ITEM %d: ",i);**

**scanf("%f",&s[i].weight);**

**printf("PROFIT OF ITEM %d: ",i);**

**scanf("%f",&s[i].profit);**

**s[i].ratio=s[i].profit/s[i].weight;**

**}**

**//DISPLAY VALUES BEFORE SORTING**

**printf("\n NUMBER\tWEIGHT\tPROFIT\tRATIO\t\n");**

**for(i=0;i<n;i++)**

**printf("%d\t%f\t%f\t%f\t\n",s[i].no,s[i].weight,s[i].profit,s[i].ratio);**

**//SORTING THE VALUES AS PER RATIO**

**for(i=0;i<n;i++)**

**{**

**for(j=0;j<i;j++)**

**{**

**if(s[j].ratio<s[j+1].ratio)**

**{**

**temp=s[j];**

**s[j]=s[j+1];**

**s[j+1]=temp;**

**}**

**}**

**}**

**//DISPLAY VALUES AFTER SORTING**

**printf("\n NUMBER\tWEIGHT\tPROFIT\tRATIO\t\n");**

**for(i=0;i<n;i++)**

**printf("%d\t%f\t%f\t%f\t\n",s[i].no,s[i].weight,s[i].profit,s[i].ratio);**

**//TO CARRY KNAPSACK OP**

**i=0;**

**totalprofit=0;**

**while(limit!=0)**

**{**

**//WHEN LIMIT IS PENDING**

**if(limit>s[i].weight)**

**{**

**s[i].frac=1;**

**limit=limit-s[i].weight;**

**totalprofit=totalprofit+s[i].profit;**

**s[i].ratio=totalprofit;**

**}**

**//WHEN LIMIT IS NOT PENDING**

**else if(limit<s[i].weight)**

**{**

**fractionalprofit=limit/s[i].weight;**

**s[i].frac=fractionalprofit;**

**totalprofit=totalprofit+fractionalprofit\*s[i].profit;**

**s[i].ratio=totalprofit;**

**limit=0;**

**}**

**i++;**

**}**

**//DISPLAY VALUES BEFORE SORTING**

**printf("\n NUMBER\tWEIGHT\tTOTAL PROFIT\tPROFIT\tRATIO\t\n");**

**for(i=0;i<n;i++)**

**printf("%d\t%f\t%f\t%f\t%f\n",s[i].no,s[i].weight,s[i].ratio,s[i].profit,s[i].frac);**

**printf("TOTAL PROFIT = %f",totalprofit);**

**}**

**EXPERIMENT 6 – 0/1 KNAPSACK**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct bag**

**{**

**float weight,ratio,profit,frac;**

**int no;**

**};**

**void main()**

**{**

**struct bag s[10],temp;**

**int i,j,n;**

**float limit,fractionalprofit,totalprofit;**

**//TAKE IN INFORMATION**

**printf("ENTER THE NUMBER OF ITEMS: ");**

**scanf("%d",&n);**

**printf("Enter the limit : ");**

**scanf("%f",&limit);**

**for(i=0;i<n;i++)**

**{**

**s[i].no=i;**

**printf("WEIGHT OF ITEM %d: ",i);**

**scanf("%f",&s[i].weight);**

**printf("PROFIT OF ITEM %d: ",i);**

**scanf("%f",&s[i].profit);**

**s[i].ratio=s[i].profit/s[i].weight;**

**}**

**//DISPLAY VALUES BEFORE SORTING**

**printf("\n NUMBER\tWEIGHT\tPROFIT\tRATIO\t\n");**

**for(i=0;i<n;i++)**

**printf("%d\t%f\t%f\t%f\t\n",s[i].no,s[i].weight,s[i].profit,s[i].ratio);**

**//SORTING THE VALUES AS PER RATIO**

**for(i=0;i<n;i++)**

**{**

**for(j=0;j<i;j++)**

**{**

**if(s[j].ratio<s[j+1].ratio)**

**{**

**temp=s[j];**

**s[j]=s[j+1];**

**s[j+1]=temp;**

**}**

**}**

**}**

**//DISPLAY VALUES AFTER SORTING**

**printf("\n NUMBER\tWEIGHT\tPROFIT\tRATIO\t\n");**

**for(i=0;i<n;i++)**

**printf("%d\t%f\t%f\t%f\t\n",s[i].no,s[i].weight,s[i].profit,s[i].ratio);**

**//TO CARRY KNAPSACK OP**

**i=0;**

**totalprofit=0;**

**while(limit>s[i].weight)**

**{**

**s[i].frac=1;**

**limit=limit-s[i].weight;**

**totalprofit=totalprofit+s[i].profit;**

**s[i].ratio=totalprofit;**

**i++;**

**}**

**//DISPLAY VALUES BEFORE SORTING**

**printf("\n NUMBER\tWEIGHT\tTOTAL PROFIT\tPROFIT\tRATIO\t\n");**

**for(i=0;i<n;i++)**

**printf("%d\t%f\t%f\t%f\t%f\n",s[i].no,s[i].weight,s[i].ratio,s[i].profit,s[i].frac);**

**printf("TOTAL PROFIT = %f",totalprofit);**

**}**

**EXPERIMENT 13 : NQUEEN BACKTRACKING PROBLEM**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<math.h>**

**int board[20],count;**

**int safety(int row,int column)**

**{**

**int i;**

**for(i=1;i<=row-1;++i)**

**{**

**if(board[i]==column)**

**return 0;**

**else**

**if(abs(board[i]-column)==abs(i-row))**

**return 0;**

**}**

**return 1;**

**}**

**void queen(int row,int n)**

**{**

**int column;**

**for(column=1;column<=n;++column)**

**{**

**if(safety(row,column))**

**{**

**board[row]=column; //no conflicts so place queen**

**if(row==n) //dead end**

**print(n); //printing the board configuration**

**else //try queen with next position**

**queen(row+1,n);**

**}**

**}**

**}**

**void print(int n)**

**{**

**int i,j;**

**printf("\n\nSOLUTION %d:\n\n",++count);**

**for(i=1;i<=n;++i)**

**printf("\t%d",i);**

**for(i=1;i<=n;++i)**

**{**

**printf("\n\n%d",i);**

**for(j=1;j<=n;++j) //for nxn board**

**{**

**if(board[i]==j)**

**printf("\tQ"); //queen at i,j position**

**else**

**printf("\t-"); //empty slot**

**}**

**}**

**}**

**int main()**

**{**

**int n,i,j;**

**void queen(int row,int n); //Looks after movement of the Queen**

**printf("\nENTER NUMBER OF QUEENS:");**

**scanf("%d",&n);**

**queen(1,n);**

**return 0;**

**}**

**EXPERIMENT 14 : GRAPH COLORING**

**#include<stdio.h>**

**#include<stdlib.h>**

**#define SIZE 10**

**int adj[SIZE][SIZE] = {0};**

**int X[SIZE] = {0};**

**void creategraph(int n)**

**{**

**int s, d, e, i, j, w;**

**for(i=1; i<=n; i++)**

**{**

**printf("Enter number edges for %d\n", i);**

**scanf("%d", &e);**

**for(j=1; j<=e; j++)**

**{**

**printf("Enter destination\n");**

**scanf("%d", &d);**

**adj[i][d] = 1;**

**}**

**}**

**}**

**int isSafe(int node, int colour)**

**{**

**int j;**

**for(j = 1; j <= node - 1; j++)**

**{**

**if(adj[node][j] == 1 && X[j] == colour)**

**{**

**return 0;**

**}**

**}**

**return 1;**

**}**

**void canColour(int node, int n, int m, int \*count)**

**{**

**int i,j,k;**

**for(i = 1; i <= m; i++)**

**{**

**if(isSafe(node, i))**

**{**

**X[node] = i;**

**if(node == n)**

**{**

**printf("The solution is\n");**

**for(k = 1; k <= n; k++)**

**{**

**printf("The colour for node %d is %d\n", k, X[k]);**

**}**

**\*count = \*count + 1;**

**printf("\n");**

**}**

**else**

**{**

**canColour(node + 1, n, m, count);**

**}}}}**

**void main()**

**{**

**int n, m, count = 0;**

**printf("Enter the number of colours: ");**

**scanf("%d", &m);**

**printf("Enter the number of vertices: ");**

**scanf("%d", &n);**

**creategraph(n);**

**canColour(1, n, m, &count);**

**printf("\nTotal solutions are %d\n", count);**

**}**

**EXPERIMENT 15: SUM OF SUBSETS**

|  |  |
| --- | --- |
| **#include<stdio.h>**  **#define max 10**  **int a[max],added[max];**  **void sos(int sum,int loc,int e,int total,int n)**  **{**  **int i;**  **if(e==sum)**  **{**  **printf("\nThe solution is : ");**  **for(i=0;i<n;i++)**  **{**  **if(added[i]==1)**  **printf("%d ",a[i]);**  **}**  **}**    **else if((e+a[loc]<=sum)&&(e+total>=sum))**  **{**  **added[loc]=1;**  **sos(sum,loc+1,(e+a[loc]),(total-a[loc]),n);**  **added[loc]=0;**  **sost(sum,loc+1,e,(total-a[loc]),n);**  **}**  **}** | **void main()**  **{**  **int i,j,temp;**  **int n,sum,total;**    **printf("Enter the number of elements : ");**  **scanf("%d",&n);**  **printf("Enter the sum : ");**  **scanf("%d",&sum);**  **total=0;**    **for(i=0;i<n;i++)**  **{**  **printf("Enter the %d element : ",i);**  **scanf("%d",&a[i]);**  **total=total+a[i];**  **}**    **for(i=0;i<n;i++)**  **{**  **for(j=0;j<n-1;j++)**  **{**  **if(a[i]<a[j])**  **{**  **temp=a[i];**  **a[i]=a[j];**  **a[j]=temp;**  **}**  **}**  **}**  **sos(sum,0,0,total,n);**  **}** |

**EXPERIMENT 16 : LONGEST COMMON SUBSEQUENCE**

**#include<stdio.h>**

**#include<string.h>**

**# define max 10**

**void main()**

**{**

**char a[max],b[max];**

**int i,j;**

**char fl[max],fw[max];**

**int row,column;**

**char f[max+1][max+1];**

**int q,w;**

**//TAKING THE INPUT**

**printf("Enter the string 1:");**

**scanf("%s",a);**

**printf("%s\n",a);**

**printf("Enter the string 2:");**

**scanf("%s",b);**

**printf("%s",b);**

**for(i=0;a[i]!='\0';i++)**

**row++;**

**for(i=0;b[i]!='\0';i++)**

**column++;**

**for(i=0;i<row+1;i++)**

**{**

**for(j=0;j<column+1;j++)**

**{**

**f[i][j]=0;**

**}**

**}**

**//MAKING THE MATRIX**

**for(i=1;i<row+1;i++)**

**{**

**for(j=1;j<column+1;j++)**

**{**

**if(a[i-1]==b[j-1])**

**f[i][j]=1+f[i-1][j-1];**

**else**

**{**

**if(f[i-1][j]>=f[i][j-1])**

**f[i][j]=f[i-1][j];**

**else if(f[i-1][j]<f[i][j-1])**

**f[i][j]=f[i][j-1];**

**}**

**}**

**}**

**//PRINTING THE MATRIX**

**printf("\nTHE MATRIX IS AS FOLLOWS\n");**

**printf(" ");**

**for(q=0;q<column+1;q++)**

**printf("%c ",b[q]);**

**for(q=0;q<row;q++)**

**{**

**printf("\n");**

**if(q>0)**

**printf("%c ",a[q-1]);**

**else**

**printf(" ");**

**for(w=0;w<column+1;w++)**

**printf("%d ",f[q][w]);**

**}**

**i=row;**

**j=column;**

**q=0;**

**while(i!=0)**

**{**

**if(f[i-1][j]>f[i][j-1])**

**{**

**if(f[i-1][j]!=f[i][j])**

**{**

**fl[q++]=a[i-1];**

**i--;**

**j--;**

**}**

**else**

**i--;**

**}**

**else if(f[i-1][j]<=f[i][j-1])**

**{**

**if(f[i][j-1]!=f[i][j])**

**{**

**fl[q++]=a[i-1];**

**i--;**

**j--;**

**}**

**else**

**j--;**

**}**

**}**

**fl[q]='\0';**

**for(i=q-1,j=0;i>=0;i--,j++)**

**fw[j]=fl[i];**

**fw[j]='\0';**

**printf("\nThe longest common subsequence : %s",fw);**

**}**